AFRL-IF-RS-TR-2000-116
Final Technical Report
August 2000

APPLYING SPECIALIZATION TO IMPROVE SURVIVABILITY OF OS KERNELS (IMMUNIX PROJECT)

Oregon Graduate Institute

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. E300

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

AFRL-IF-RS-TR-2000-116 has been reviewed and is approved for publication.

APPROVED: [illegible]

THOMAS F. LAWRENCE
Project Engineer

FOR THE DIRECTOR: [illegible]

WARREN H. DEBANY, JR., Technical Advisor
Information Grid Division
Information Directorate

APPLYING SPECIALIZAITON TO IMPROVE SURVIVABILITY OF OS KERNELS (IMMUNIX PROJECT)

Calton Pu
Crispin Cowan
Virgil Gligor
Heather Hinton
Jonathan Walpole

Contractor: Oregon Graduate Institute.
Contract Number: F30602-96-1-0331
Effective Date of Contract: 27 August 1996
Contract Expiration Date: 26 August 1999
Short Title of Work: Applying Specialization to Improve Survivability of OS Kernels (IMMUNIX PROJECT)
Period of Work Covered: Aug 96 – Aug 99

Principal Investigator: Calton Pu
Phone: (503) 690-1214
AFRL Project Engineer: Thomas F. Lawrence
Phone: (315) 330-2925

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and was monitored by Thomas F. Lawrence, AFRL/IFGA, 525 Brooks Road, Rome, NY.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE AUGUST 2000	3. REPORT TYPE AND DATES COVERED FINAL (AUG 96 - AUG 99)

4. TITLE AND SUBTITLE APPLYING SPECIALIZATION TO IMPROVE SURVIVABILITY OF OS KERNELS (IMMUNIX PROJECT)	5. FUNDING NUMBERS C - F30602-96-1-0331 PE - 62301E PR - D985 TA - 02 WU - 04
6. AUTHOR(S) Calton Pu, Crispin Cowan, Virgil Gligor, Heather Hinton, and Jonathan Walpole	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Oregon Graduate Institute, 20000 NW Walker Road, Beaverton, OR 97006 Ryerson Polytechnic Institute, 350 Victoria Street, Toronto, Ontario, Canada M5B 2K3 VDG Inc., 6009 Brookside Drive, Chevy Chase, MD 20815	8. PERFORMING ORGANIZATION REPORT NUMBER N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) DARPA/ITO 3701 N. Fairfax Drive Arlington, VA 22203 — AFRL/IFGA 525 Brooks Road Rome, NY 13441-4505	10. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-IF-RS-TR-2000-116
11. SUPPLEMENTARY NOTES AFRL Project Engineer: Thomas F. Lawrence, IFGA, (315) 330-2925	
12a. DISTRIBUTION AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The results are summarized for the DARPA contract"Applying Specialization to Improve Survivability of OS Kernels" (Immunix Project). The main objective of the Immunix Project is to improve the survivability of operating system (OS) kernels against security attacks. The important problem to be solved is to be able to survive attacks that exploit unknown vulnerabilities in system software. The primary method is to use specialization techniques and toolkit developed in the Synthetix project. The key idea is to use the specialization toolkit to make the system adaptive with respect to security threats. Examples include restricting the system to prevent it entering states characterized by attacks, dynamic response to detected intrusion, dynamic change of previously granted privileges, and using run-time specialization to generate a large number of correct variants of many OS modules, so some of the variants will be resistant to new, previously unknown attacks.

14. SUBJECT TERMS Computer Security, Operating System Kernel, Malicious Attack, Intrusion Detection, Adaptivity			15. NUMBER OF PAGES 32
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

Table of Contents

1 Objectives and Results

The main objective of the Immunix Project is to improve the survivability of operating system (OS) kernels against security attacks. The important problem to be solved is to be able to survive attacks that exploit *unknown* vulnerabilities in system software. We call these unknown vulnerabilities *security faults*. The problem of surviving unknown vulnerabilities is particularly difficult because of the complex space of consequences that can result from an unknown bug. This problem is particularly important to solve because a very large portion of security problems result from unknown (or unpatched) security vulnerabilities.

The primary method is to use the *specialization* techniques and toolkit developed in the Synthetix project (DARPA/ONR grant N00014-94-1-0845). Section 1.1 explains the Synthetix specialization concepts and techniques. Section 1.2 explains how specialization concepts pertain to enhancing system survivability against security faults. Section 1.3 describes how we used these ideas to develop a variety of tools for enhancing system survivability.

1.1 Immunix BackGround: Synthetix

Synthetix sought to enhance system performance through specialization. "Specialization" is normally thought of as "optimization using invariants." However, an operating system is both general purpose (supporting a variety of hardware configurations and application workloads) and long-running (nearly infinite intended running time). Thus very few conditions can be considered invariant, and operating system implementations are forced to consider a multitude of conditions at many points in execution.

Synthetix introduced the notion of a *quasi-invariant*: a condition that is "nearly" invariant, in that it holds for a long time, if not for ever [22]. Having identified a quasi-invariant condition, system specialization requires an OS developer to perform two kinds of operations:

Specialize Components: Here the implementer optimizes some component with respect to the quasi-invariant condition. The optimization can be static, where the implementer enumerates all likely conditions and prepares an optimized component for each condition, or dynamic, where the system emits optimized code at run time to take advantage of conditions as they arise. In either case, the specialization is *incremental*, in that enabling and changing of specialized components occurs at run time.

The optimization can be done by hand, or can be automated using a partial evaluation tool [2]. Our experience indicates that both forms of specialization are effective: specialization by hand enables powerful optimizations and is fairly easy [22], automatic specialization has also been shown to bring substantial performance benefits [28].

Guard the Conditions: By definition, quasi-invariants are subject to change from time to time. When a quasi-invariant changes, the specialized components that depended on the *old* value of the quasi-invariant condition become *invalid*, and must be replaced with either generic components, or differently specialized components. *Guarding* is the monitoring of system state variables that embody quasi-invariant conditions such that the system is notified when a quasi-invariant condition becomes false.

The Synthetix project sought to ease these tasks by providing the following *specialization tools*:

Tempo: A partial evaluation compiler for C code. Tempo can perform static specialization, where all of the invariants and their values are known at compile time, and also dynamic specialization, where the variables to be made invariant are specified at compile time, but the *values* for those variables is specified at *specialization time* during the program's execution.

TypeGuard & MemGuard: Guarding is primarily accomplished by inserting "guarding code" into the program at all places where program activity may invalidate a quasi-invariant condition. TypeGuard and MemGuard are tools to facilitate guarding by identifying places in the program where guarding code needs to be inserted. TypeGuard is a compile-time tool that identifies all statements in a C program that effect writes to values of a certain "type" (really, member of a struct) and thus identify places in the program where guarding code needs to be inserted. MemGuard is a run time tool that uses virtual memory to detect writes to quasi-invariant condition terms, and thus identify locations in the program that are invalidating quasi-invariant conditions.

Specialization Classes: Specialization Classes is a framework for binding quasi-invariants and specialized components together [5, 27].

1.2 Specialization for System Survivability

One of the key characteristics of incremental specialization is its dynamic modification of the control flow of a program. In Synthetix we used this technique to improve the program's performance, and importantly, we ensured that the program's functionality was not affected by doing the specialization in a controlled way. Specifically, we asserted invariants and added guarding code to ensure that they were preserved. In this approach you can view the triggering of a guard as a signal that "it is no longer safe to run this code".

An observation about security attacks such as buffer overflow attacks is that they also work by dynamically modifying the control flow of a program. Hence, you can view them as dynamic specialization attempts. However, a key difference is that they are not performed in a controlled way with respect to preserving the original functionality of the program (of course!). One way to view the use of stack guard is that it is a tool for inserting some piece of state (the canary), asserting it to be an invariant of the program, and inserting guarding code to keep track of it and ensure that execution does not continue when the invariant ceases to hold. Thus Immunix sought to exploit two key properties of this technique to enhance system survivability:

Compatibility: Specialized components use variant implementations, and thus security attacks that depend on implementation properties may not be "compatible" with specialized components.

Restrictions: The guarding code that detects violations of quasi-invariant conditions can also be used to detect security attacks that violate desired security properties.

We generalized these concepts to consider a broad spectrum of ways that existing software can be *adapted* to survive security faults [7]. We categorize adaptations in terms of *what* is adapted, and *how* it is adapted:

What is Adapted: "What" is adapted is either a component's *interface* or its *implementation*. Naturally, what is "interface" and what is "implementation" is a relativistic view: an adaptation is an "interface" adaptation if it affects *other* components, and it is an "implementation" adaptation if the adaptation has no externally visible effects other than reduced vulnerability.

How It is Adapted: "How" a component is adapted to reduce vulnerability is either a *restriction* or an *obfuscation*. A "restriction" is an adaptation that characterizes some behaviors as "bad", and *a priori* prohibits or prevents those behaviors.An "obfuscation" radomizes some aspect of the component while obscuring the current configuration from would-be attackers. This makes it difficult for an attacker to deploy an attack(s), as the configuration details required for the attack cannot be `reliably`and repeatedly predicted by the attacker.

Table 1: Security Bug Tolerance Techniques

	Interface	Implementation
Restriction	• File system access controls • Firewalls • TCP Wrappers • Java sandbox • TCP SYN time out adjustment	• *Small* TCB & code removal, i.e. bastion hosts • Static Type checking • Dynamic Checking: array bounds checking, assertion checking, Stack-Guard, non-executable segments
Obfuscation	• Winnowing and Chaffing • Deception Toolkit	• Random TCP initial sequence number • Random code or data layout • Random QoS change

The two values for each of the two dimensions produce a quadrant of security bug tolerance adaptations. Table 1 shows this quadrant, populated with example security bug tolerance techniques. Some cells in the quadrant are old ("well-understood") and thus heavily populated with known examples and techniques, while others are relatively new and unexplored. Immunix research resulted in both advances in various quadrants of the grid [11, 1, 4] as well as a greater understanding of the relative strengths and weaknesses of the quadrants [10, 9, 7].

1.3 Immunix Results

In view of the categorization of survivability enhancements, Immunix sought to enhance system survivability by exploring each of the quadrants of the grid. Sections 1.3.1 through 1.3.4 describe our experiments in each quadrant. Our contributions to the grid of survivability adaptations is shown in Table 2.

Table 2: Immunix Security Bug Tolerance Techniques

	Interface	Implementation
Restriction	• SubDomain • GuardHouse	• StackGuard • SAM • VDG Code Analysis Tool
Obfuscation	• Morphing File System	• StackGuard "diversity" canary

1.3.1 Interface Restrictions

We developed two interface restriction technologies: *SubDomain* and *GuardHouse*.

SubDomain: SubDomain is a technique to enhance system survivability through the creative application of the classical technique of domain enforcement. SubDomain provides an *enhancement* to the host system's access control model. The enhancement allows the system administrator to clearly and concisely specify the domain (the set of resources) that a *program* may access. We use this facility to confine programs that are privileged but not necessarily trusted to a small security domain. By constraining the resources accessible to such programs we can significantly enhance the survivability of servers that run these potentially vulnerable programs. SubDomain is an interface restriction in the sense that it restricts the interface between programs and the file system.

The SubDomain kernel extension is operational, can constrain the domain of a native program, and is running in production on Crispin Cowan's workstation/notebook computer. In conjunction with VDG, we have written a paper describing the innovative properties provided by SubDomain [4]. SubDomain exhibits two novel and unique properties:

1. SubDomain can effectively contain the potential security vulnerabilities imposed by threaded execution and loadable module extensions, such as the `mod_fp` module that provides support for Microsoft's Front Page Extensions in the Apache web server.
2. SubDomain can exhibit performance *improvements*. SubDomain exploits security specification information for *prefetching* data, so as to improve the performance of a confined application. This is in stark contrast to the usual security mechanism, which measures performance impacts entirely in terms of added costs.

GuardHouse: GuardHouse is a kernel enhancement to detect and reject Trojan Horse programs through cryptographic signatures. The system operator cryptographically signs all of the programs on the system that need to run at a given trust level (e.g. programs that need to run as root). The kernel checks the cryptographic signatures of programs that try to start at this trust level, and rejects those programs that either don't have a signature, or have an incorrect signature.

To better understand the purpose of GuardHouse, consider the attacker's tools RootKit and Back Orifice [30]. Once an attacker has gained privilege on a host, they most often install *back door* programs that permit them easy access to the host in the future, and *Trojan Horse* programs that make it difficult for the system administrator to detect the back door. Rootkit is a collection of back door and Trojan Horse programs for UNIX, and takes the form of modified versions of programs like **`login`** and **`sshd`** for the back door, and modified versions of **`ps`** and **`ls`** to make it difficult to detect the back door. "Back Orifice" is a similar package of back doors and Trojan Horses for Windows, with the added feature that installing Back Orifice causes an announcement to be made on an IRC channel.

GuardHouse detects and rejects the installation of these back doors and Trojan Horses in that the attacker cannot install Trojans without knowing the system administrator's private key, which is stored off line and only pulled off the shelf to sign newly installed legitmate programs. The attackers programs will thus not bear the required signature, and the kernel will refuse to run these Trojaned programs as root. Instead, the programs will abort, and the system administrator will *immediately* learn that critical software components have been corrupted, and appropriate action can be taken. In any case, the attacker does not gain the privileges intended by the corrupted programs.

Similar to SubDomain, GuardHouse is an interface restriction in that it restricts the interface between programs and the kernel, imposing an additional restriction upon programs that try to execute with strong privileges.

1.3.2 Implementation Restrictions

Implementation restriction techniques have been our most successful area of investigation. We have three discrete implementation restriction projects: *StackGuard*, *SAM*, and the VDG code analysis tool.

StackGuard: StackGuard [11] is a compiler method for protecting vulnerable programs against "stack smashing" buffer overflow attacks [21, 20, 26, 14]. StackGuard produces protected programs by emitting code to instrument the execution stack of the running program, to detect when a attack has been attempted. When StackGuard detects such an attack, it causes the application to exit, rather than yield control to the attacker. StackGuard is implemented as a small enhancement to the `gcc` compiler, and is intended to protect `SetUID` privileged programs, and other programs run by `root`.

StackGuard has been our most significant result. Buffer overflow attacks constitute a majority of software security vulnerabilities [13, 12], and the "stack smashing" variety treated by StackGuard is the most common form of buffer overflow attack. StackGuard offers substantial compatibility advantages, which we demonstrated by using StackGuard to protect an entire Linux distribution of programs [3]. Thus StackGuard has the potential to substantially reduce the vulnerability of commonly used systems.

We have used StackGuard to protect an entire Linux distribution [3], and released both the compiler and the protected distribution on the web [18, 19]. These software releases have produced substantial technology transfers, as detailed in Section 4.5.1. Thousands of individuals have visited these web sites, and StackGuarded systems are known to be in production use at a variety of security-sensitive sites.

The protected system was also submitted for testing to the Lincoln/AFRL intrusion detection competition. The intrusion detection relevance is that stack smashing attacks against StackGuard-protected programs produce highly reliable intruder alerts. The Lincoln/Rome test measured many parameters, but two key tests were the false-positive reporting rate, and the ability of intrusion detection systems to detect "new, novel" attacks: attacks that are *unknown* to the intrusion detection system. While StackGuard is limited to detecting buffer overflow attacks, it is *unique* in its ability to detect unknown attacks, and in its false-positive rate of *zero*.

SAM (Security Adaptation Manager): Security is often skimped because of the performance, compatibility, and administrative overheads that it may impose. More over, some security mechanisms can be *adjusted* to trade off the security offered against the costs imposed. For instance, firewall rules can be programmed to be permissive (convenient but insecure) or strict (inconvenient but more secure). Both StackGuard and CoDomain have multiple "modes" of operation, where one mode offers better protection, and another mode offers better performance. In addition, the higher-performance modes have the property that initial attempts to penetrate the system are likely to set off intrusion alerts.

SAM was designed to leverage these situations by providing *adaptive* protection in response to intrusion detection data [17]. A system will (mostly) run in a higher-performance mode. SAM will switch, or adapt, the system to a higher-security mode when attempted attacks are detected. Different levels of protection are possible, with the higher-security modes implemented when more attacks are detected over a predetermined time interval. We provide default values for the number of attacks observed and the time interval size; these values can be adjusted for individual installations. SAM is preconfigured to allow three different levels of protection (referred to as protection postures) for the overall system. How these protection postures are configured is definable by the user.

SAM was implemented and tested with StackGuard intrusion alarms. This mechanism can be easily configured to response to intrusions detected by other intrusion detection systems, or as reported by frameworks such as CIDF.

SAM was primarily developed by our subcontractors at Ryerson, leveraging the Adaptation Space tools developed at OGI under the Heterodyne project (DARPA grant F40602-96-1-0302).

VDG Penetration Analysis Tool: The VDG penetration analysis tool is used to restrict those implementations that contain flaws which could enable the operating system to be penetrated. By writing penetration-resistance specifications, the code is statically analyzed to ensure that none of the five previously identified properties of penetration-resistant systems [15] is violated. It is important to note that this tool operates on the code itself, and therefore directly ensures that the code is correct with respect to the desired properties. We feel that this is much more satisfactory than ensuring that the algorithm or process is correct and then trying to ensure that the code is a correct implementation of the algorithm. Any implementation that is not compliant is not used until it can be made compliant.

The VDG penetration analysis tool was used to analyze portions of a Linux operating system. The results of this investigation, as well as a collection of known flaws and vulnerabilities from CERT advisories and rootshell archives, was used to classify potential flaws. We developed a classification system based on flaw "accessibility", which is the ease of exploitation, and the "exposure", which is a measure of the potential damage caused. We also developed a metric for measuring flaws based on their "region of influence." This incorporates the observation that systems with a higher level of security may be immune to certain exploits, whereas systems whose security is at or below a certain threshold will have their security degraded to a lower level. Since systems already operating below this degraded level will not be affected, we say that only those systems with a specified security level within the region of influence (bounded by the threshold level from above, and the degraded level below) are affected.

1.3.3 Interface Obfuscations

We found it difficult to construct *effective* interface obfuscations. Our one effective interface obfuscation technique was the Morphing File System (MFS) a defense that we designed and implemented to protect sensitive files (e.g. /etc/passwd) from attack by *renaming* them to something obscure, and only "telling" the true name to programs that need to access these sensitive files. All other programs trying to find these sensitive files must search through a "forrest" of fake files, trying to determine which is the real one. MFS is an interface obfuscation in that the name of the object to be accessed has been obscured, and only clients knowing the true name can access the object.

While the MFS was "successful" in that it did make it hard for attacking programs to find the sensitive files, it was also complex and brittle. The MFS achieves *precisely* the semantics of Levitt's PACLs (Program Access Control Lists) [29]: disclosing the true name of an obscured file to a program is identical to adding the program to the ACL. PACLs are simpler to administer, and thus more likely to be administered correctly. PACLs can be implemented more simply, and thus are more likely to be implemented correctly. Thus for controlling program access to file system resources, interface restrictions seem to be more cost-effective than interface obfuscations.

1.3.4 Implementation Obfuscations

We found implementation obfuscations to be even more difficult to make effective than interface obfuscations. Implementation obfuscation techniques consist primarily of randomly altering the memory layout of programs so as to frustrate attacks that depend on the specific location of certain key objects in memory. Such attacks are almost universally buffer overflow techniques. For instance, Forrest proposed a compiler enhancement that introduced random amounts of padding to stack frames, making it difficult for attackers to precisely aim their buffer overflow attacks.

The problem with memory obfuscation techniques is that attackers have developed *adaptive* attack methods that do not *need* to know the precise layout of memory [21, 14, 20, 26] limiting the effectiveness of this technique. For instance, an attacker does not need to know the precise offset of a buffer containing attack code; the attacker can prepend a string of NOP instructions in front of the attack code, and "lob" flow control into the midst of this field of NOPs.

Our work in implementation obfuscation comprised an enhancement to the StackGuard "terminator canary" mechanism [6]. A vulnerability arose, allowing attackers to undetectably corrupt stack frames protected with the "terminator" style of StackGuard protection. We investigated adding "jitter" to the position of the terminator canary, but found this to be difficult because the gcc compiler assumes a *fixed* offset between the stack frame and the local automatic variables. Use of StackGuard's "random canary" was found to be both more effective and more efficient.

1.4 Immunix Results Summary

Our goal was to enhance the survivability of operating systems against security attacks that exploit *unknown* security bugs, using the specialization techniques developed in the Synthetix project. We have achieved this goal in the following ways:

- Constructed a categorization of possible *security bug tolerance* techniques [10, 9].
- Populated this categorization with new security bug tolerance techniques in each quadrant of the grid (sections 1.3.1 through 1.3.4).
- One of these techniques (StackGuard [11]) is strikingly effective in protecting large numbers of systems against large classes of security vulnerabilities [3] and has been widely adopted by the Linux user community.

2 Chronological List of Publications

1. "Specialization Classes: An Object Framework for Specialization", by Crispin Cowan, Andrew Black, Charles Krasic, Calton Pu, Jonathan Walpole, Charles Consel, and Eugen-Nicolae Volanschi. In the proceedings of the International Workshop on Object Orientation in Operating Systems (IWOOOS'96), Seattle, WA, October 1996 [5].
2. "A Specialization Toolkit to Increase the Diversity of Operating Systems", by Calton Pu, Andrew Black, Crispin Cowan, and Jonathan Walpole. In the proceedings of the ICMAN Immunity-Based Systems Workshop, Nara, Japan, December 1996 [23].
3. "Microlanguages for Operating System Specialization", by Calton Pu, Andrew Black, Crispin Cowan, Jonathan Walpole, and Charles Consel. In the proceedings of the SIGPLAN Workshop on Domain-Specific Languages, Paris France, January 1997 [24].
4. "Declarative Specialization of Object-Oriented Programs", by Eugen N. Volanschi, Charles Consel, Gilles Muller, and Crispin Cowan. In the proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'97), Atlanta, GA, October 1997 [27].
5. "StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks", by Crispin Cowan, Calton Pu, David Maier, Heather Hinton, Peat Bakke, Steve Beattie, Aaron Grier, Perry Wagle, and Qian Zhang. In the proceedings of the 7th USENIX Security Symposium, San Antonio, TX, January 1998 [11].
6. "Death, Taxes, and Imperfect Software: Surviving the Inevitable", by Crispin Cowan, Calton Pu, and Heather Hinton. In the proceedings of the New Security Paradigms Workshop, Charlottesville, VA, September 1998 [10].
7. "Survivability From a Sow's Ear: The Retrofit Security Requirement", by Crispin Cowan and Calton

Pu. In the proceedings of the 1998 Information Survivability Workshop, Orlando, FL, October 1998 [9].

8. "Protecting Systems from Stack Smashing Attacks with StackGuard", by Crispin Cowan, Steve Beattie, Ryan Finnin Day, Calton Pu, Perry Wagle, and Erik Walthinsen. In the proceedings of the 1999 Linux Expo, Raleigh, NC, May 1999 [3].
9. "GuardHouse: Locking the Stable Door Ahead of the Trojan Horse", by Steven M. Beattie, Andrew P. Black, Crispin Cowan, Calton Pu, and Lateef P. Yang, in preparation [1].
10. "SubDomain: Extracting Performance from Fine-Grained Security Mechanisms", by Crispin Cowan, Steve Beattie, Calton Pu, Perry Wagle, and Virgil Gligor, November 1999, in preparation [4].
11. "SAM: Security Adaptation Manager", by Heather M. Hinton, Crispin Cowan, Lois Delcambre, and Shawn Bowers. In the proceedings of the Annual Security Applications Conference, Phoenix, AZ, December 1999 [17].
12. "Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade", by Crispin Cowan, Perry Wagle, Calton Pu, Steve Beattie, and Jonathan Walpole. In the proceedings of the DARPA DISCEX conference on Information Survivability, Hilton Head Island, SC, January 2000 [12]. This paper is also an invited talk at System Administration & Network Security (SANS 2000), Orlando, FL, March 2000.
13. "The Cracker Patch Choice: An Analysis of *Post Hoc* Security Techniques", by Crispin Cowan, Heather Hinton, Calton Pu, and Jonathan Walpole, October 1999, submitted for review [7].
14. "A Difficulty-Opportunity of Attack (DOA) Assessment of Retro-Fit Security Mechanisms", by Heather Hinton and Crispin Cowan, October 1999, in preparation [16].

3 Project Personnel

The Immunix project comprised the OGI prime contractor, described in Section 3.1, and two subcontractors, described in sections 3.2 and 3.3.

3.1 OGI: Prime Contractor

The principle investigator of Immunix was Calton Pu. The co-investigative faculty were:

- Crispin Cowan (technical lead)
- Charles Consel
- Dylan McNamee
- Jonathan Walpole

Immunix technical staff at OGI were:

- Perry Wagle (full-time lead developer)
- Erik Walthinsen (partially assigned to Immunix)
- Ryan Finnin Day (partially assigned to Immunix)
- Peter Bakke (intern)
- Aaron Grier (intern)
- Tim Chen (intern)

- Lateef Yang (intern)

Immunix graduate students at OGI were:

- Steve Beattie: lead developer of SubDomain, M.Sc. earned
- Qian Zhang: developed MemGuard

3.2 Ryerson Polytechnic University: Subcontractor

The lead investigator of the Ryerson subcontract was Heather Hinton. Ryerson also employed one student: James Wysynski.

3.3 VDG, Inc.: Subcontractor

The lead investigator of the VDG, Inc. subcontract was Virgil Gligor. VDG also employed Kevin Hildebrand, Bob Fourney, and Randall Winchester, all on a part time basis.

4 Interactions

4.1 Formal Presentations

1. "Specialization Classes: An Object Framework for Specialization", presented by Crispin Cowan at the International Workshop on Object Orientation in Operating Systems (IWOOOS'96), Seattle, WA, October 1996 [5].
2. "A Specialization Toolkit to Increase the Diversity of Operating Systems", presented by Calton Pu at the ICMAN Immunity-Based Systems Workshop, Nara, Japan, December 1996 [23].
3. "Declarative Specialization of Object-Oriented Programs" presented by Eugen N. Volanschi at the Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'97), Atlanta, GA, October 1997 [27].
4. "StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks" presented by Crispin Cowan at the 7th USENIX Security Symposium, San Antonio, TX, January 1998 [11].
5. Crispin Cowan, "StackGuard 1.1: Stack Smashing Protection for Shared Libraries", brief presentation and poster at the 1998 IEEE Symposium on Security and Privacy, Oakland, CA, May 3-6, 1998 [11].
6. "Death, Taxes, and Imperfect Software: Surviving the Inevitable" presented by Crispin Cowan at the New Security Paradigms Workshop, Charlottesville, VA, September 1998 [10].
7. "Survivability From a Sow's Ear: The Retrofit Security Requirement" presented by Crispin Cowan at the 1998 Information Survivability Workshop, Orlando, FL, October 1998 [9].
8. "Protecting Systems from Stack Smashing Attacks with StackGuard" presented by Crispin Cowan at the 1999 Linux Expo, Raleigh, NC, May 1999 [3].
9. "SAM: Security Adaptation Manager", to be presented by Heather M. Hinton at the Annual Security Applications Conference, Phoenix, AZ, December 1999 [17].

4.2 Invited Talks, Informal Presentations and Panel Appearances

1. Charles Consel visited Xerox PARC (Palo Alto, California, 2/21/97), Microsoft (Redmond, Washington, 2/24/97), University of Washington (Seattle, Washington, 2/26/97), and gave talks on the Tempo Specializer and system support for microlanguages.

2. Crispin Cowan visited Intel's Data Security Group (Hillsboro, Oregon, 3/12/97) to give a talk on "Immunix: Survivability through Specialization".
3. Calton Pu was an invited panelist in the panel on "Survivability in the Face of Malicious Attack", in the Sixth IFIP Working Conference on Dependable Computing for Critical Applications (DCCA-6), panel chaired by Teresa Lunt of DARPA/ITO, March 1997.
4. Crispin Cowan presented "Immunix: Adaptive System Survivability" to the DARPA QUORUM OS Security Workshop, Washington, DC, July 25, 1997. The presentation included a live demonstration of the Morphing File System.
5. Crispin Cowan presented "Immunix: Adaptive System Survivability" to the DARPA Security Wrappers and Composition Workshop, Lake Tahoe, CA, August 13 - 16, 1997. The presentation included live demonstrations of the StackGuard and the Morphing File System.
6. Crispin Cowan presented "Immunix: Adaptive System Survivability" as an invited talk to the University of Toronto Electrical and Computer Engineering department, October 10, 1997.
7. Crispin Cowan presented "System Security: Threats and Counter-measures" to the ADP Dealer Services Group in Portland, Oregon, November 7, 1997. This talk was an introduction to survivability techniques for the turn-key systems that ADP markets to auto dealerships.
8. Crispin Cowan presented "Immunix: Adaptive System Survivability" to TEM (the Technology Exchange Meeting), Utica, NY, December 2 - 3, 1997. The presentation included live demonstrations of the StackGuard and the Morphing File System.
9. Crispin Cowan, "StackGuard: Automatic Adaptive Detection and Prevention of Buffer Overflow Attacks", DARPA Intrusion Detection Joint PI Meeting, Annapolis, MD, February 4, 1998.
10. Calton Pu, "Applying Specialization to Improve Survivability of Operating Systems", at the DARPA Adaptive Architecture Workshop, Palo Alto, CA, May 12-13, 1998.
11. Calton Pu presented a an invited talk at the HASE'98 (High Assurance System Engineering) symposium, November 13-14, 1998, Washington, DC, entitled "System Survivability Through Security Bug Tolerance." The talk presents an overview of our survivability research, enhancing the awareness of the high assurance research community of the survivability problem.
12. Crispin Cowan presented "StackGuard: Recent Impact and Current Developments", with Calton Pu, at the December 1998 DARPA Intrusion Detection PI Meeting, December 8-10, 1998, Lexington, MA [8].
13. "SAM: Security Adaptation Manager" presented by Crispin Cowan and Heather Hinton at the August 2 - 6, 1999 DARPA Joint Intrusion Detection and Information Assurance PI Meeting, Phoenix, AZ.
14. "Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade", to be presented by Crispin Cowan at System Administration & Network Security (SANS 2000), Orlando, FL, March 2000. This talk is based on our DISCEX paper [12].

4.3 Meetings & Committee Memberships

This section lists various meetings and collaborations. We explicitly list meetings with external organizations, but not meetings between OGI and subcontractors, or between OGI and DARPA.

Regarding meetings with DARPA, the PI, Prof. Calton Pu, has maintained regular meeting with DARPA/ITO managers. He has visited DARPA regularly (no less than once every 6 months) and has met with Dr. Teresa Lunt, Dr. Robert Laddaga, and Dr. Gary Koob to report on the progress of the project. Crispin Cowan also visited Dr. Lunt, Dr. Koob, and Dr. Laddaga on several occasions. This is in addition to PI meetings and workshops.

Regarding internal meetings between OGI and the Ryerson and VDG subcontractors, both Prof. Pu and Crispin Cowan met with the subcontractors on numerous occasions. Professor Hinton and Professor Gligor both visited OGI on two occasions each. Each of these visits advanced one or more of the papers jointly produced between OGI and the subcontractors [4, 7, 10, 11, 16, 17].

1. Crispin Cowan visited CMU, Prof. Daniel Jackson, on December 9th, 1996, to discuss collaboration. We discussed the use of `lackwit`, their software analysis tool, for the static verification of guarded code. The OGI group sent real code to be analyzed by the CMU group for evaluation of the tool robustness for production code.
2. Calton Pu and Crispin Cowan met with David Tennenhouse during his visit to OGI on February 5 & 6, 1998. Professor Pu briefed Dr. Tennenhouse on Immunix's approach to the survivability problem, and Crispin Cowan demonstrated the unique intrusion detection result produced by StackGuard: a "false-positive" intrusion report rate of *zero.*
3. Crispin Cowan met with Joonees Chay of WireX Communications on February 10, 1998, to discuss technology transfer opportunities.
4. Crispin Cowan becomes a member of the Board of Directors of WireX Communications, July 1998.
5. Crispin Cowan met with Robert Laddaga and Gary Koob at DARPA headquarters on September 22, 1998 to discuss project progress, including demonstrating an operational StackGuard + CoDomain computer, and demonstrating the machine's ability to tolerate various generic classes of security bugs.
6. Crispin Cowan becomes CTO (Chief Technologies Officer) of WireX Communications, October 1998.
7. Calton Pu attended the Symposium on Reliable Distributed Systems (SRDS) as a member of the program committee, West Lafayette, IN, October 21-23, 1998.
8. Crispin Cowan and Calton Pu met with Peter G. Neumann and several other interested parties including Paul Walczak (ARL) and Brian Randell (U. Newcastle upon Tyne) to discuss the innovative survivability properties of the *open source* movement. Cryptographers and security researchers have long known the value of public review of an *algorithm* or *design*. Here we hypothesize that the same security benefits may be conferred through public review of the source code of an *implementation*. The discussion produced a mailing list and a nascent community to advocate among the open source community for survivability enhancements, and among the survivability community for the use of open source products and practices. This community is called *Robust Open Source.*
9. Crispin Cowan met with Virgil Gligor (Immunix subcontractor at VDG, Inc.) and Peter G. Neumann on December 10, 1998, in Lexington, MA, to discuss on-going developments in the *Robust Open Source* community.
10. Calton Pu attended the International Conference on Data Engineering (ICDE), March 23-26, 1999, in Melbourne, Australia, as the program committee co-chair and a member of the steering committee.
11. Crispin Cowan attended the program committee meeting for the 8th USENIX Security Symposium, April 19, 1999, Burlington MA. Professor Cowan shepherded the paper "Synthesizing Fast Intrusion Prevention/Detection Systems from High-Level Specifications" [25] which was an information survivability project. This paper was very well received by the conference audience.

12. Calton Pu met with Joonees Chay (WireX Communications) to discuss Immunix technology transfer to WireX, Portland, OR, July 1999.

13. Crispin Cowan participated in the DARPA/NSA/ARL invitational workshop on Insider Misuse, August 16 - 18, Santa Monica, CA, hosted by Dick Brackney (NSA) and Peter Neumann (SRI). Professor Cowan's participation focussed on the "prevention" aspect of insider misuse through the pervasive application of information survivability techniques. The workshop report is here: `http://www2.csl.sri.com/insiders/`

14. Crispin Cowan visited Lt.Col. Paul Walczak (Army Research Lab) August 24, 1999, in Arlington, VA, to discuss information survivability results as they pertain to the ARL's mission.

15. Crispin Cowan attended the 8th USENIX Security Symposium, August 25 - 26, 1999, in Washington, DC. Professor Cowan attended as a member of the program committee, and chaired the track on "Cages": process confinement techniques that enhance the survivability of host operating systems.

4.4 Courses

Crispin Cowan developed three related courses on information survivability using experience derived from the Immunix project:

CSE 585: System Survivability: Staying Available in a Hostile World. This was a graduate seminar in survivability issues. This course aims to educate students in state-of-the-art tools and techniques for enhancing the survivability of computer systems under attack. The course was offered in the winter quarter of 1998, attracting 6 students. The home page for the course is here:
`http://www.cse.ogi.edu/~crispin/585/`

CSE 527: Principles and Practices of System Security. This is a recurring graduate course that tries to span the bridge between security theory and security practice. Students are given an introduction to the principles of secure system design, and then asked to evaluate actual common systems in the context of the principles of security. The course is offered annually, and attracted 12 students in its first offering. The home page for the course is here:
`http://www.cse.ogi.edu/~crispin/527/`

Site Security: Firewalls and Beyond. This one-day industrial continuing education course is a condensed version of CSE 527. Like the graduate course, it introduces students to the theory of secure systems, and evaluates current host security products in light of that theoretical understanding.

4.5 Technology Transfer

Immunix technology transfer has been highly effective, through two different channels: open source distribution, and explicit transfer of technology to WireX Communications, Inc.

4.5.1 Open Source Distribution

Since we publicly released the StackGuard compiler and the Linux systems we have protected with StackGuard [18, 19] they have become very popular products. As with all open source products, it is difficult to accurately gauge the number of copies in use. However, we have gathered the following anecdotes & statistics:

- Approximately 10,000 unique visitors have browsed the Immunix web sites [18, 19]
- Approximately 500 copies of the StackGuard compiler have been downloaded

- Approximately 200 copies of the complete StackGuarded Linux operating systems have been downloaded.
- StackGuard is in use by a Florida law enforcement agency: Ty Roden, Computer Services Supervisor, Lake Country Sheriff's Office, (352)343-9500, `http://www.lcso.org`.
- StackGuard is in use by the Verio web hosting company: Stacey Son, CTO & co-founder, Verio, `sson@verio.net`.

On October 28, 1998, attackers broke into `rootshell.com` (a full-disclosure hacker web site) using a previously unknown buffer overflow vulnerability to SSH (a popular cryptographically strong authentication and communication package). The threat implied by attackers who could hack a hacker site caused great concern, and the threat implied by an *unknown* vulnerability to a very common strong authentication package caused greater concern.

Fortunately, it was exactly this kind of problem that StackGuard was designed to defend against: unknown vulnerabilities. On November 4th, two "celebrity" hackers ("Aleph One" who moderates the Bugtraq security mailing list, and Alan Cox, a renowned Linux developer) advocated the use of StackGuard to defend against this unknown vulnerability. The immediate result was several thousand new hits to the StackGuard home page. We continue to receive requests for information, enhancements, and ports to this day.

4.5.2 Transfers to WireX Communications, Inc.

WireX Communications, Inc. (`http://wirex.com`) is in the business of building highly survivable network server appliances. An essential property of an "appliance" is that it does not need regular maintenance. Thus the usual approach of quickly distributing and applying patches when vulnerabilities are discovered is no longer appropriate. Information survivability technologies such as Immunix that enable a server to resist attacks, even in the presence of unknown security bugs, bring information appliances much closer to the goal of a truly maintenance-free appliance.

In July 1999, WireX and OGI were jointly awarded a DARPA/SBIR/STTR contract to transfer technology from OGI to WireX by applying Immunix tools to a workgroup server. WireX is incorporating and distributing Immunix technologies in the following ways:

- WireX is hosting the `http://immunix.org` site, where on-going maintenance & support of Immunix tools is being provided to the community, largely under the GPL license. `http://immunix.org` has received approximately 3000 visitors since it was announced October 1, 1999.
- All C code in WireX appliances is protected with StackGuard.
- WireX server appliances incorporate the SubDomain kernel extension, and most of the services installed on WireX server appliances are confined with SubDomain.

A substantial portion of the Immunix team also transferred from OGI to WireX at the conclusion of the Immunix project. Crispin Cowan is now the full-time CTO of WireX. Steve Beattie received his M.Sc. from OGI, and now is a senior developer with WireX. Peter Bakke, who interned with Immunix in summer 1997, has been a full-time developer at WireX since February 1999.

5 Discoveries

Immunix has resulted in two theoretical discoveries:

1. That survivability adaptations can be effectively categorized according to *what* they adapt (in-

terfaces or implementations) and *how* they are adapted (restrictions or obfuscations) [10, 9].

2. While the techniques in all of the quadrants in the 2x2 grid of survivability adaptations can be effective, we discovered that some of the quadrants are more cost-effective than others. Specifically, restrictions appear to be more cost-effective than obfuscations [7].

Immunix also resulted in the following practical discoveries:

1. That elegantly designed restrictions can be effective in hardening large bodies of software against large classes of security vulnerabilities, without sacrificing compatibility or performance [11, 6, 12].
2. That determining which components of a system are security-critical is problematic. If a sufficiently transparent facility is available (i.e. StackGuard) it is both simpler and more effective to just protect *everything* in the system [3].
3. That re-building any large system completely from source is intrinsically difficult, even without the trivial incompatibilities introduced by StackGuard.

References

[1] Steven M. Beattie, Andrew Black, Crispin Cowan, Calton Pu, and Lateef Yang. GuardHouse: Locking the Stable Door Ahead of the Trojan Horse. Submitted for review, May 1999.

[2] Charles Consel and Francois Noĭl. A General Approach to Run-time Specialization and its Application to C. In *23rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'96)*, St. Petersburgh Beach, FL, January 1996.

[3] Crispin Cowan, Steve Beattie, Ryan Finnin Day, Calton Pu, Perry Wagle, and Erik Walthinsen. Protecting Systems from Stack Smashing Attacks with StackGuard. In *Linux Expo*, Raleigh, NC, May 1999.

[4] Crispin Cowan, Steve Beattie, Calton Pu, Perry Wagle, and Virgil Gligor. SubDomain: Extracting Performance from Fine-Grained Security Mechanisms. Submitted for review, May 1999.

[5] Crispin Cowan, Andrew Black, Charles Krasic, Calton Pu, Jonathan Walpole, Charles Consel, and Eugen-Nicolae Volanschi. Specialization Classes: An Object Framework for Specialization. In *Proceedings of the Fifth International Workshop on Object-Orientation in Operating Systems (IWOOOS '96)*, Seattle, WA, October 27-28 1996.

[6] Crispin Cowan, Tim Chen, Calton Pu, and Perry Wagle. StackGuard 1.1: Stack Smashing Protection for Shared Libraries. In *IEEE Symposium on Security and Privacy*, Oakland, CA, May 1998. Brief presentation and poster session.

[7] Crispin Cowan, Heather Hinton, Calton Pu, and Jonathan Walpole. The Cracker Patch Choice: An Analysis of Post Hoc Security Techniques. Submitted for review, October 1999.

[8] Crispin Cowan and Calton Pu. StackGuard: Recent Impact and Current Developments. DARPA Intrusion Detection System PI Meeting, December 1998.

[9] Crispin Cowan and Calton Pu. Survivability From a Sow's Ear: The Retrofit Security Requirement. In *Proceedings of the 1998 Information Survivability Workshop*, Orlando, FL, October 1998. `http://www.cert.org/research/isw98.html`.

[10] Crispin Cowan, Calton Pu, and Heather Hinton. Death, Taxes, and Imperfect Software: Surviving the Inevitable. In *Proceedings of the New Security Paradigms Workshop*, Charlottesville, VA, September 1998.

[11] Crispin Cowan, Calton Pu, Dave Maier, Heather Hinton, Peat Bakke, Steve Beattie, Aaron Grier, Perry Wagle, and Qian Zhang. StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks. In *7th USENIX Security Conference*, pages 63–77, San Antonio, TX, January 1998.

[12] Crispin Cowan, Perry Wagle, Calton Pu, Steve Beattie, and Jonathan Walpole. Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade. In *DARPA Information Survivability Conference and Expo (DISCEX)*, January 2000. `http:// schafercorp-ballston.com/discex`.

[13] Michele Crabb. Curmudgeon's Executive Summary. In Michele Crabb, editor, *The SANS Network Security Digest*. SANS, 1997. Contributing Editors: Matt Bishop, Gene Spafford, Steve Bellovin, Gene Schultz, Rob Kolstad, Marcus Ranum, Dorothy Denning, Dan Geer, Peter Neumann, Peter Galvin, David Harley, Jean Chouanard.

[14] "DilDog". The Tao of Windows Buffer Overflow. `http://www.cultdeadcow.com/cDc_files/cDc-351/`, April 1998.

[15] Sarbari Gupta and Virgil D. Gligor. Towards a Theory of Penetration-Resistant Systems and Its Application. In *IEEE Computer Security Foundations Workshop*, Franconia, NH, June 1991.

[16] Heather Hinton and Crispin Cowan. A Difficulty-Opportunity of Attack (DOA) Assesment of Retro-Fit Security Mechanisms. Submitted for review, March 1999.

[17] Heather M. Hinton, Crispin Cowan, Lois Delcambre, and Shawn Bowers. SAM: Security Adaptation Manager. In *Annual Computer Security Applications Conference (ACSAC)*, Phoenix, AZ, December 1999.

[18] Immunix. Adaptive System Survivability. `http://www.cse.ogi.edu/DISC/projects/immunix`, 1997.

[19] Immunix. Survivable Operating Systems and Components. `http://immunix.org`, 1999.

[20] "Mudge". How to Write Buffer Overflows. `http://l0pht.com/advisories/ bufero.html`, 1997.

[21] "Aleph One". Smashing The Stack For Fun And Profit. *Phrack*, 7(49), November 1996.

[22] Calton Pu, Tito Autrey, Andrew Black, Charles Consel, Crispin Cowan, Jon Inouye, Lakshmi Kethana, Jonathan Walpole, and Ke Zhang. Optimistic Incremental Specialization: Streamlining a Commercial Operating System. In *Symposium on Operating Systems Principles (SOSP)*, Copper Mountain, Colorado, December 1995.

[23] Calton Pu, Andrew Black, Crispin Cowan, and Jonathan Walpole. A specialization toolkit to increase the diversity of operating systems. In *Proceedings of the 1996 ICMAS Workshop on Immunity-Based Systems*, Nara, Japan, December 1996.

[24] Calton Pu, Andrew Black, Crispin Cowan, Jonathan Walpole, and Charles Consel. Microlanguages for Operating System Specialization. In *SIGPLAN Workshop on Domain-Specific Languages*, Paris, France, January 1997.

[25] Ravi Sekar and Prem Uppuluri. Synthesizing Fast Intrusion Prevention/Detection Systems from High-Level Specifications. In *8th USENIX Security Symposium*, Washington, DC, August 1999.

[26] Nathan P. Smith. Stack Smashing vulnerabilities in the UNIX Operating System. `http://millcomm.com/ nate/machines/security/stack-smashing/nate-buffer.ps`, 1997.

[27] Eugen N. Volanschi, Charles Consel, Gilles Muller, and Crispin Cowan. Declarative Specialization of Object-Oriented Programs. In *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'97)*, Atlanta, GA, October 1997.

[28] Eugen-Nicolae Volanschi, Gilles Muller, and Charles Consel. Safe Operating system Specialization: The RPC Case Study. In *Proceedings of the First Annual Workshop on Compiler Support for System Software*, Tuscon, AZ, February 1996.

[29] D.R. Wichers, D.M. Cook, R.A. Olsson, J. Crossley, P. Kerchen, K. Levitt, and R. Lo. PACL's: An Access Control List Approach to Anti-viral Security. In *Proceedings of the 13th National Computer Security Conference*, pages 340–349, Washington, DC, October 1-4 1990.

[30] "ISS X-Force". ISS Vulnerability Alert: Windows Backdoors Update. Bugtraq mailing list, `http://geek-girl.com/bugtraq/`, September 10 1998. Back Orifice.

DISTRIBUTION LIST

addresses	number of copies
AFRL/IFGA ATTN: TOM LAWRENCE 525 BROOKS ROAD ROME, NEW YORK 13441-4505	10
OREGON GRADUATE INSTITUTE 20000 NW WALKER ROAD BEAVERTON, OR 97006	5
AFRL/IFOIL TECHNICAL LIBRARY 26 ELECTRONIC PKY ROME NY 13441-4514	1
ATTENTION: DTIC-OCC DEFENSE TECHNICAL INFO CENTER 8725 JOHN J. KINGMAN ROAD, STE 0944 FT. BELVOIR, VA 22060-6218	1
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY 3701 NORTH FAIRFAX DRIVE ARLINGTON VA 22203-1714	1
ATTN: NAN PFRIMMER IIT RESEARCH INSTITUTE 201 MILL ST. ROME, NY 13440	1
AFIT ACADEMIC LIBRARY AFIT/LDR, 2950 P.STREET AREA B, BLDG 642 WRIGHT-PATTERSON AFB OH 45433-7765	1
AFRL/MLME 2977 P STREET, STE 6 WRIGHT-PATTERSON AFB OH 45433-7739	1

AFRL/HESC-TDC 2698 G STREET, BLDG 190 WRIGHT-PATTERSON AFB OH 45433-7604	1
ATTN: SMDC IM PL US ARMY SPACE & MISSILE DEF CMD P.O. BOX 1500 HUNTSVILLE AL 35807-3801	1
TECHNICAL LIBRARY D0274(PL-TS) SPAWARSYSCEN 53560 HULL ST. SAN DIEGO CA 92152-5001	1
COMMANDER, CODE 4TL000D TECHNICAL LIBRARY, NAWC-WD 1 ADMINISTRATION CIRCLE CHINA LAKE CA 93555-6100	1
CDR, US ARMY AVIATION & MISSILE CMD REDSTONE SCIENTIFIC INFORMATION CTR ATTN: AMSAM-RD-OB-R, (DOCUMENTS) REDSTONE ARSENAL AL 35898-5000	2
REPORT LIBRARY MS P364 LOS ALAMOS NATIONAL LABORATORY LOS ALAMOS NM 87545	1
ATTN: D'BORAH HART AVIATION BRANCH SVC 122.10 FOB10A, RM 931 800 INDEPENDENCE AVE, SW WASHINGTON DC 20591	1
AFIWC/MSY 102 HALL BLVD, STE 315 SAN ANTONIO TX 78243-7016	1
ATTN: KAROLA M. YOURISON SOFTWARE ENGINEERING INSTITUTE 4500 FIFTH AVENUE PITTSBURGH PA 15213	1

USAF/AIR FORCE RESEARCH LABORATORY AFRL/VSOSA(LIBRARY-BLDG 1103) 5 WRIGHT DRIVE HANSCOM AFB MA 01731-3004	1
ATTN: EILEEN LADUKE/D460 MITRE CORPORATION 202 BURLINGTON RD BEDFORD MA 01730	1
OUSD(P)/DTSA/DUTD ATTN: PATRICK G. SULLIVAN, JR. 400 ARMY NAVY DRIVE SUITE 300 ARLINGTON VA 22202	1